Neural Radiance Fields for 3D Scene Reconstruction:

A Comparative Study of Advanced Models

1st Dung Vu
MSc in ACIT
Data Science
OsloMet
Oslo, Norway
duvu35218@oslomet.no

2nd Reynato Jr. Matencio
MSc in ACIT
Data Science
OsloMet
Oslo, Norway
remat0831@oslomet.com

3rd Thomas Nygaard
MSc in ACIT
Data Science
OsloMet
Oslo, Norway
thnyg9592@oslomet.no

***Abstract*—Neural Radiance Fields (NeRF) has revolutionized 3D scene representation and rendering that offers realism and efficiency. This paper conducts and implements several advanced NeRF models: Kilo-NeRF concentrates more on leveraging multi-level perceptron (MLP) to depicts geometric appearance scene; X-NeRF highlights the process called "'Cross-Spectral" to improve the camera's angle across various ranges during the exploration and exploitation process, namely "Normalized Cross Device"; PlenOctrees's paradigm is to enhance the speed on the rendering process by forecasting a spherical harmonic scene that automatically eliminates the view direction as an input; Lastly, Instant-NGP model involves an efficiency and effectiveness by reducing the training times and its performanced.**

***Index Terms*—Machine Learning, 3D-Scene, NeRF, Kilo-NeRF, X-NeRF, PlenOctrees**

I. Introduction

75 years ago, the first image was taken from the Polaroid camera [1]. Hence, it has begun to explore and revolutionize an instant record 3D world in a 2D image. Nowadays, due to the advancement of artificial intelligence (AI), it enforced to simulate how a single light work in the actual world or a method also known as *inverse rendering*. It empowers the researchers to reconstruct a 3D scene that is inspired and originally from the images that were collected from the different angles [2].

Neural Radiance Fields (NeRF) is one of the foremost models that was developed to generate a 3D environment that can capture intricate geometric shapes and textures from complex images whilst utilizing less disk space [3]. NeRF's purpose was to originally overcome problem vision synthesis. This kind of task requires generating a 3D scene or object from the gathered photos that has taken from diverse angles and viewpoints [4].

In the area of computer vision and photogrammetry, high-quality 3D reconstruction is a pivotal discipline that comprises various applications, for instance, digital preservation, quality checking, reverse engineering, structural monitoring, etc [5]. Nonetheless, over the past few years, different key factors, such as low-cost, portable, and flexible 3D measuring strategies, has been significant in providing a high standard geometric accuracy and high resolution ins and outs. Different extant approach for 3D reconstruction is considered as widely categorized as whether it is contact or non-contact techniques [6].

In the area of computer visions, computer graphics, and virtual/augmented reality, the differentiable rendering approach has gained a lot of attention in the research. This process can be comprehended as a function that allocates the variables of geometries, materials, lights, and cameras to the pixels' concentration; for instance, data flows from the scene parameters to the photo. Nonetheless, the primary goal of differentiable rendering is to spread the picture's pixel gradients to the scene [7].

Artificial intelligence and machine learning techniques have been implemented in computer graphics to simplify some complex computations and procedures. Machine learning algorithms are essential for methods like ray tracing for the purpose of flexibility in various scenarios. Promisingly, some methods may perform better, but it's often machine learning that accomplishes better [8]

II. Literature Review

Right after Mildenhall et al. [9] successful research on NeRF in 2021, numerous researchers works have been conducted that are relevant to the initial NeRF including Mip-NeRF [10] that particularly concern on designing a NeRF model that exists in a multi-level representation and antialiasing; Block-NeRF [11] that is mainly concerned by applying an adaptable big scene; Mip-NeRF [12] which focuses on generating stereo and panoramic scenes or objects. *Figure 2* represents the visual development of dynamic NeRF between 2021-2023.

The Time Development Chart (TDC) is divided into 3 main components that implicated the revolution between

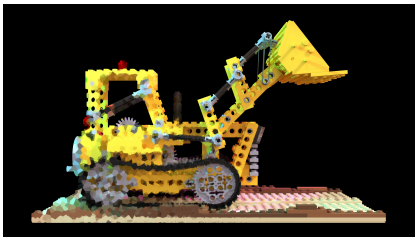


Fig. 1. 3D model example [9]

2021-2023. 2021 is meant for green, yellow for 2022, and pink for 2023.

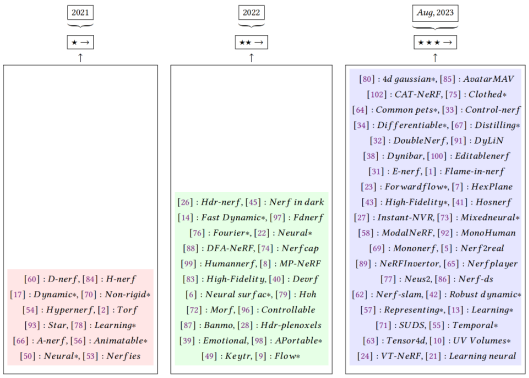


Fig. 2. Time Chart Development of Dynamic-NeRF [13]

Inside these boxes, the researchers contributions are included within the item numbers and acronyms. In 2021, the data showed the lowest research investigation, which was the first mark of NeRF 2021. A year later, it depicts significant growth in the discipline. In the late summer of August 2023, revealed an exponential trend in research interest and results [13].

The continuous interest on Dynamic NeRF shows an undeniably potential to improve more or totally substitute the conventional approach in rendering. As a result, its development is prone for more complex area of research and innovation in the area of 3D modeling and rendering [13].

In order to apprehend the full potential and function of NeRF, it is pivotal to understand the essence of radiance. Generally, radiance is the measurement of light quantity that either penetrates or exits inside a space at a specific angle. Radiance has 3 elements which represents the hues of Red, Green and Blue when this data is captured in RGB [3]. There are factors that are considered in radiance point of space.

- Source of light that illuminates that point
- The availability of space (or the volume density) that mirrors light at point
- The surface's textural features

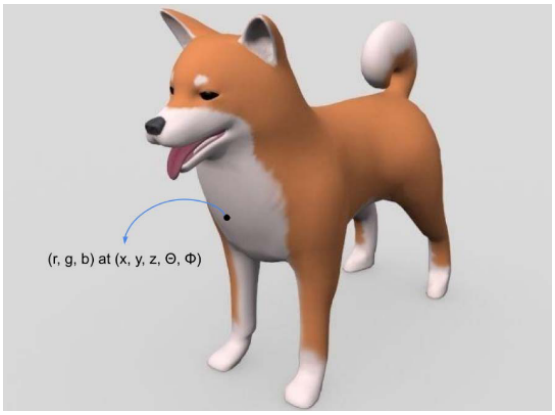


Fig. 3. The radiance (r, g, b) at a point (x, y, z) [3]

The *Figure 3* illustrates the radiance value in specific point in 3D scene in a different point of viewpoint. All of this radiance value and locations inside the three-dimensional scene constitute the radiance field [3].

Normally, polygon meshes or voxel grids are applied for storing 3D scenes. In contrast, preserving the voxels is largely costly. Nevertheless, only the solid surfaces can depict polygon's meshes, that is why it is not applicable for medical imaging [4].

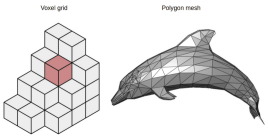


Fig. 4. The radiance (r, g, b) at a point (x, y, z) [4]

A. Related Work

The Neural Redience Field (NeRF) is a pretty new concept in the ground of Deep Learning (DL) and computer vision. It was first implemented in 2020 by ECGV paper *"NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis"* that received an ("outstanding recognition for best papers"). As a result, the method has been widely utilized and obtained almost 800 citation so far. [9], [14]. Below are the following studies that has been implemented on the various type of NeRF model.

1) ***Kilo-NeRF****:* A study from [15] found out that one of the biggest challenge in NeRF is its long rendering and training times. Even if the devices are embedded with multi-GPU cluster, rendering requires happening in real time on the consumer's devices. Therefore, this study will focus on cumulative NeRF's rendering speed. *Figure 5* depicts Multilevel Layer Perceptron's (MLP) high capacity, rather than implementing a one representation.

NeRF leveraging Multi-level Perceptron (MLP) to illustrate the geometric and appearance of the scene. This model is sampled hundreds of thousands of times for millions of pixels in the rendering volumetric process. Although the MLP process may have a deep and wide structure, it performs very slowly. In order to shorten up the rendering process, the hidden unit layers must decrease. To address this, we must deploy a multitude of small, separate networks that individually represent a portion of the scene [15].

Despite its wide and promising performance and the fact that Kilo-Nerf is capable of generating 800 x 800 medium of quality images through interactive frame rates, the speedups are still inconsistent to develop full HD images in real time. Reducing the smaller networks might aid by expanding the method and achieve a fast result. Nevertheless, if this executes improperly, the aftermath might lead to a higher consumption of storage [15]. Memory efficient data structures might lessen, which enables networks to be constructed closely to the surfaces [16], [17].

2) ***X-NeRF****:* One study proposed by [18] a *X-NeRF*, a process to adapt a ***"Cross-Spectral"*** scene that represents a given image captured from various spectrum sensitivity from the camera, that was originally applied to neural radiance fields formulations. It improves camera postures across different ranges during the process and exploits ***"Normalized Cross Device"*** (NXDC) to render photos from diverse condition to the arbitrary viewpoints that are symmetric on the same resolution.

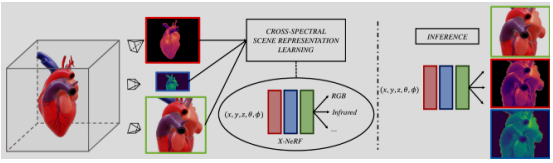


Fig. 5. X-NeRF Rendering [18]

As described by [18]. There are two shortcomings that are considered in this mysterious setting. First were the precise camera poses for the images that were fetched from the various cameras to train NeRF. Although these details are achieved effortlessly dealing with RDG images. The second issue is related to the variance between sensors and resolutions that correlates to the field of view (FoV). This accounts when throwing rays across 3D rays scope to reassure that traced matching pixel in each camera would exactly on the location. For instance, to deploy processing forward-facing scenes from ***Normalized Device Coordinates*** convention went unsuccessful on this [19], [20].
Overall, the research demonstrated the accuracy of the process to produce high-quality images from the different modalities of X-NeRF on images that was achieved in multi-spectral rig, thanks to NXDC space [18]. *Figure 7* shows the different camera centers in 3D spectrum in 50, 500 and 5000 epochs.

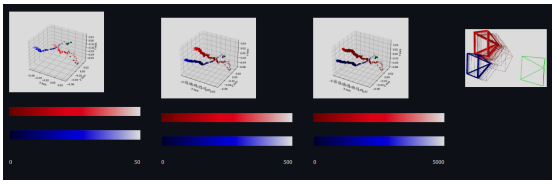


Fig. 6. Relative cameras positions during training [18]

Right after it reached around 250 epochs, approximately 7.5 minutes, the camera became consistent.

3) ***PlenOctreess****:* Another interesting study from [21] introduced an innovative *Neural Radiance Fields* (NeRFs) in the actual time using ***"PlenOctrees"***. The primary focus of this paradigm is to enhance the efficiency and speed the rendering of NeRF. The research has implemented 800 x 800 images with 150 FPS, that is typically 3000 much faster in comparison with traditional NeRFs.

By pretabulating the NeRF onto a PlenOctree enables real-time performance. Leveraging the closed-form spherical basis function will factorize the appearance and maintain view dependencies, effect like peculiarities. In general, it is feasible to train NeRFs to forecast a spherical harmonic representation of radiance via eliminating the view direction as an input to the neural network [21].

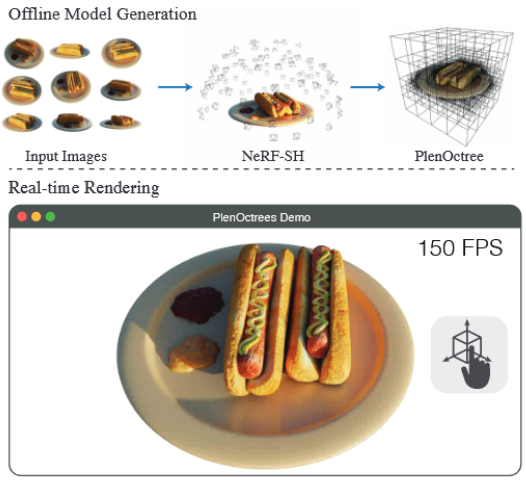


Fig. 7. Real-time NeRF with PlenOctrees [21]

Figure 8 illustrates a series of images of a scene that generates a 3D volumetric model that can be rendered in real-time. They introduced PlenOctrees that can captivate view-dependent, relying on the effect of specularities. Conventional NeRF renders slower compare to this rendering image of magnitude [21].

Additionally, they demonstrate that PlenOctress may be adapted to decrease the reconstruction loss even more, as a

result of equal level or even much better quality unlike from the competing approaches. Lastly, this kind of approach could potentially decrease time in training optimally. Therefore, there, it is unnecessary to wait for a long time for the NeRF training to full congregate. Application such as *6-DOF* industrial and product visualization together with next generation *AR/VR* systems might pave the way in real-time neural rendering approach [21].

4) ***Instant-NGP****:* The advancement of technology in computer vision became far blurry between the reality and simulation, yet promising. This improvement can automate the physical world with these cutting-edge technologies that enables turning the projection in virtual world reality. Nowadays, researchers rely on in triangulation of light to generate a 3D model. However, the reality is that we have different angles of perspective of the scene, and parse those distinct model that allows for novel viewpoints rather than the original input image [8]. Therefore, this method paved the way the researchers into streamlining the NeRF's process direct to Instant Neural Graphic Primitives (Instant-NGP)

One of the groundbreaking studies in NeRF is from [22]. This creative study showcases how possible it is to fill an instant with high-quality results from the different tasks using neural networks whilst sustaining simplicity and efficiency. They demonstrated how a single GPU could run multiple tasks. A 2D surface is utilized by the zero-level set of a signed distance function (SDF) that adapts in 3D space. NeRF [19] readjust 2S images and its camera position in a volumetric radiance and density field that is projected using ray marching.

III. Methodology

The study will be deeply concentrating on the following existing models based on the extant literature from several online databases. This allows the researchers to delve technically by accessing and gathering the open source code to evaluate rendering efficiency, identifying the various models, compare, and check the scalability of each model.

Each model was reviewed and deployed based on their perspective process to be able to fulfill a real time 3D images.

I wrote this part please do not citate anything here [18]

A. How Nerf work for 3D reconstruction?

Neural Radiance Fields (NeRF) represent a novel approach to 3D object reconstruction and view synthesis. The method encodes a scene as a continuous 5D function, mapping 3D spatial coordinates (x, y, z) and 2D viewing directions (,) to RGB colors and volume density. This function is approximated by a Multi-Layer Perceptron (MLP), which is trained on a set of input images with known camera poses. The process starts with the (x, y, z) coordinates are first processed through 8 fully-connected layers with ReLU activations.Then it produces the volume density and a 256-dimensional feature vector. The feature vector is then concatenated with the viewing direction d and passed through an additional fully-connected layer. The final layer outputs the view-dependent RGB color.

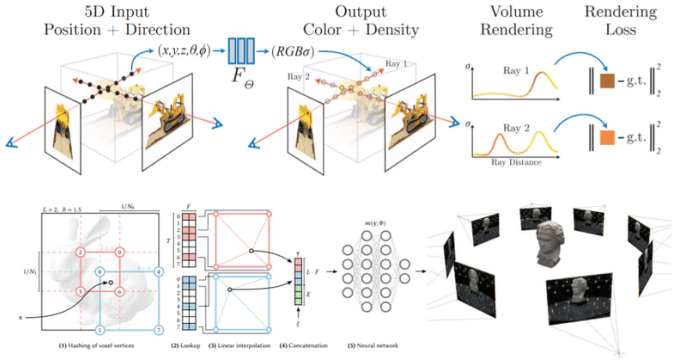


Fig. 8. How Neural Radiance Fields (NeRF) and Instant Neural Graphics Primitives work [23]

Specifically, to render novel views, NeRF employs volume rendering techniques: it casts rays through each pixel, samples points along these rays, and uses the MLP to predict colors and densities at these points. These predictions are then accumulated using a discretized version of the volume rendering equation to produce the final pixel colors.

To enhance the model's ability to represent high-frequency details, NeRF applies a positional encoding to the input coordinates before feeding them into the MLP. This encoding maps the inputs to a higher-dimensional space using sinusoidal functions. The training process optimizes the MLP by minimizing the difference between rendered and ground truth images across multiple views. This approach allows NeRF to learn a compact neural representation of the 3D scene, capturing both structure and view-dependent appearance, and enables the synthesis of high-quality novel views from arbitrary camera positions [11].

1) Frames per second-FNS: is the number of distinct sequential photos that are played in a single second of video playback. The measurement can be leveraged for any medium of image that ought to employ an illusion of movement. It can be measured through an individual frame that is captivated or illustrated per second [24].

Low or High concentration can depend upon the situation and viewer's expectation. Generally, 60 FPS is considered as high while 30 FPS beyond is considered as low [24].

2) PNSR: A signal measurement that compares the given or processed signal from its original signal is called peak signal-to-noise ratio (PSNR). It has been used to regulate threshold level and the amount of thresholds in multi-level segmentation [25].

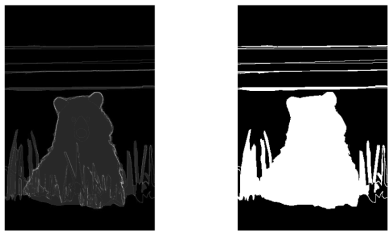
Fig. 9. Low vs High FPS [24]

A case study from [25] evaluated the PSNR to obtain a reliable analytic method for image segmentation algorithms. The experiments used the database named **Berkeley BSR300**. It contains 300 various images that comprise different types of scenes.

(a) (b)

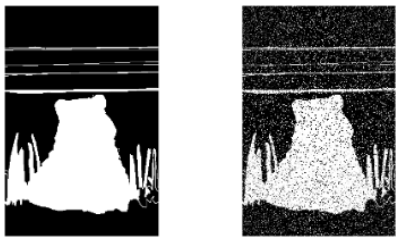
Fig. 10. Database picture **(a)** and its respective ground truth **(b)** [26]

Even if the PSNR can provide a promising measure of quality, it does not always match what other folk sees. Measurement like Structural-Similarity Index Measure (SSIM) could are combined alongside side with PSNR for a thorough assessment [27].



(a) (b)

Fig. 11. Automatic filled ground truth image [25]



(a) (b)

Fig. 12. Binary mask B **(a)** & bad Segmentation mask B after salt and pepper noise **(b)** [25]

B. How 3D-models applied in real-time application?

1) Instant NeRF: The research explored one of the prominent paradigms in NeRF. Primarily, it underscored and illustrated three-dimensional (3D) objects within the idea of the industrial metaverse. Intentionally, it aimed to reach a high-fidelity structure of objects in a virtual setting. The study deployed several helping equipments, for example, Unity, Photon Pun2 and Oculus Interaction SDK to enhance captivating metaverse. The figure below is am example of interaction in the field of metaverse in Valencia city [28].

(a) **(b)**

Fig. 13. Industrial Meteverse [28]

Additionally, the study provided a step-by-step flowchart manual for converting existing models into a more immersive industrial metaverse. Utilizing this NeRF technology gives a new possibility for more 3D presentation, collaboration, and exploration.

2) Vanilla NeRF: Neural Radiance Fields (NeRF) presents a novel approach to scene representation through a

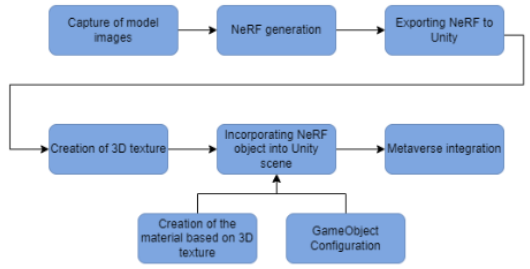


Fig. 14. Workflow of Industrial Metaverse [28]

continuous volumetric function encoded in a neural network. The fundamental innovation lies in representing a static scene as a continuous five-dimensional function that maps spatial coordinates and viewing directions to volume density and directional emitted radiance, enabling high-fidelity novel view synthesis.

Second, the framework implements a differentiable volume rendering system based on classical techniques. This rendering approach accumulates color and density values along camera rays through numerical quadrature, enabling direct optimization from two-dimensional images without requiring three-dimensional supervision. The system employs stratified sampling along rays, maintaining the continuous nature of the scene representation while enabling efficient rendering.

Third, NeRF introduces a positional encoding scheme that maps input coordinates to a higher-dimensional space through sinusoidal functions. This transformation enables the network to represent high-frequency variations in scene properties, addressing the traditional limitation of neural networks in learning high-frequency functions. The encoding applies to both spatial coordinates and viewing directions, significantly improving the reproduction of fine geometric details and view-dependent effects.

Fourth, the optimization framework combines these elements through a straightforward training procedure that requires only a set of input images with known camera poses. This approach minimizes the error between rendered and observed images through gradient descent, effectively learning to reconstruct complex scene geometry and appearance without additional supervision or loss terms.

3) KILO NeRF: KiloNeRF introduces an innovative approach to accelerate Neural Radiance Fields through a divide-and-conquer strategy. The primary innovation lies in decomposing the scene representation into multiple independent smaller neural networks, each responsible for modeling a specific spatial region within the scene volume. This architectural design fundamentally differs from the original NeRF's single network approach.

The methodology employs five key technical innovations. First, the scene decomposition strategy subdivides the bounded scene volume into a uniform three-dimensional grid, where each cell is assigned its own Multi-Layer Perceptron (MLP). This spatial partitioning enables each network to specialize in representing a localized region, significantly reducing the complexity required for each individual network while maintaining high-fidelity scene representation.

Second, KiloNeRF implements an efficient knowledge distillation framework for training. The process begins with training a teacher network using the original NeRF architecture, followed by transferring this knowledge to the multiple smaller networks. This distillation approach prevents artifacts in free space and ensures consistent scene representation across the grid cells, while the subsequent fine-tuning on training images allows each network to specialize in its local region.

Third, the framework incorporates an optimized rendering pipeline through empty space skipping and early ray termination techniques. This sampling strategy efficiently allocates computational resources by focusing only on regions containing visible scene content and terminating ray traversal when sufficient opacity is accumulated, enabling real-time rendering performance.

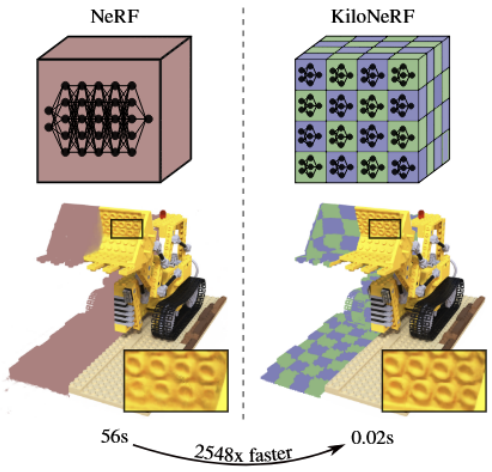


Fig. 15. NeRF vs Kilo-NeRF [15]

Fourth, the methodology introduces a novel view-dependent modeling approach through regularization. By applying targeted regularization to the network layers responsible for view-dependent effects, KiloNeRF maintains high visual quality while ensuring stable view-dependent rendering, despite using smaller networks.

Fifth, the framework employs an advanced sampling strategy combining stratified sampling with an occupancy grid structure. This hybrid approach enables efficient ray traversal through the scene volume while maintaining rendering quality,

particularly in regions with complex geometry or appearance variations.

Compared to Vanilla NeRF's monolithic architecture, KiloNeRF's distributed approach presents significant advantages for real-time 3D applications. While Vanilla NeRF requires extensive computation through a deep network for each spatial query, KiloNeRF's smaller networks, combined with its efficient spatial partitioning and optimized sampling strategies, enable substantially faster rendering without compromising visual quality. The knowledge distillation framework ensures that this speed improvement doesn't sacrifice the high-fidelity reconstruction capabilities of the original NeRF, while the regularized view-dependent modeling maintains consistent appearance across viewpoints. This architectural redesign fundamentally addresses the computational bottleneck of neural radiance fields, making it particularly suitable for interactive 3D applications requiring real-time performance.

Fig. 16. Architecture Comparison: Vanilla NeRF (Left) vs KiloNeRF (Right) Neural Network Layer Structures [15]

4) FAST NeRF: FastNeRF introduces an innovative approach to neural radiance field rendering by fundamentally restructuring the representation and computation pipeline for real-time performance. The methodology centers on a graphics-inspired factorization of the rendering process, enabling dramatic speed improvements while maintaining high visual fidelity.

The core innovation lies in the factorized architecture that decomposes the traditional 5D neural radiance field into two separate functions. The first function processes only spatial position information to produce a compact deep radiance map, while the second function handles viewing directions to generate corresponding weights. This architectural separation enables independent processing of position and direction-dependent features, fundamentally transforming the rendering pipeline's efficiency.

The methodology implements three key technical innovations. First, the position-dependent network generates a compact deep radiance map parameterized by components that capture the spatial complexity of the scene. Second, the direction-dependent network produces weights that, when combined with the radiance map through an inner product operation, efficiently reconstruct view-dependent effects. Third, the framework introduces an intelligent caching mechanism that leverages the factorized architecture to store pre-computed values, dramatically reducing runtime computational requirements.

The rendering process employs a novel optimization strategy that combines cached positions and directions through efficient inner product operations. This approach enables real-time rendering by replacing expensive network evaluations with rapid cache lookups and simple mathematical operations. The system maintains high visual quality through careful calibration of the cache resolution and component dimensionality, effectively balancing memory usage with rendering fidelity.

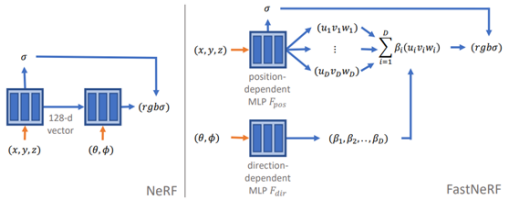


Fig. 17. Network Architecture Comparison: Vanilla NeRF (Left) vs FastNeRF (Right) Feature Processing Pipeline [29]

Compared to Vanilla NeRF, FastNeRF introduces a revolutionary redesign of neural radiance field rendering that fundamentally transforms the computational approach. While Vanilla NeRF requires evaluating a deep neural network hundreds of times along each ray to process combined 5D inputs (position and direction), resulting in millions of network evaluations per image, FastNeRF's factorized architecture separates this problem into two distinct components: a position-dependent function and a direction-dependent function. This architectural distinction manifests in several key technical advantages. First, where Vanilla NeRF must process high-dimensional inputs through multiple dense layers to capture view-dependent effects, FastNeRF's separated computation allows for efficient caching of spatial information in a compact deep radiance map while independently handling directional effects through a smaller network. Second, FastNeRF replaces Vanilla NeRF's computationally expensive volume rendering process, which requires repeated neural network evaluation, with an efficient inner product operation between cached radiance maps and direction-dependent weights. Third, while Vanilla NeRF's memory requirements scale with the product of spatial and directional dimensions, FastNeRF's factorized approach allows for independent scaling of spatial and directional components, dramatically reducing memory requirements while maintaining rendering quality. This fundamental reimagining of the rendering pipeline enables FastNeRF to overcome Vanilla NeRF's computational bottleneck, achieving real-time rendering rates through cache lookups and efficient mathematical operations rather than intensive neural network evaluations.

5) PlenOctrees: PlenOctrees introduces an innovative approach to neural radiance field rendering by transforming the continuous neural representation into an efficient octree-based volumetric structure while preserving view-dependent effects. The methodology fundamentally restructures the rendering pipeline to achieve real-time performance without sacrificing visual quality.

The core innovation lies in the hierarchical representation using view-dependent volumetric elements. The framework employs three key technical breakthroughs. First, it introduces a novel spherical harmonic representation for encoding view-dependent effects, eliminating the need for viewing direction as a neural network input. Second, it develops an octree-based spatial hierarchical structure that efficiently stores and queries the scene's geometric and appearance information. Third, it implements a direct optimization strategy for the octree structure itself, enabling quality improvements without requiring full neural network convergence.

The methodology employs a sophisticated training process that operates in two phases. Initially, the system trains a modified NeRF network to predict spherical harmonic coefficients rather than raw RGB values. Subsequently, this neural representation is transformed into a PlenOctree structure, where each leaf node stores both density values and spherical harmonic coefficients. This design enables efficient rendering through direct evaluation of the basis functions, eliminating the need for costly neural network inference during rendering.

The rendering process leverages the hierarchical nature of the octree structure to efficiently traverse and accumulate radiance values. The system combines rapid octree traversal with efficient spherical harmonic evaluation, enabling real-time rendering through a purely computational approach that eliminates the need for neural network queries during inference.

When compared to Vanilla NeRF's continuous neural representation, PlenOctrees demonstrates significant advantages for real-time 3D applications through several key architectural innovations. First, while Vanilla NeRF requires hundreds of neural network evaluations per ray to capture view-dependent effects, PlenOctrees' spherical harmonic representation analytically encodes view dependency, enabling direct computation of appearance from any viewpoint without network inference. This transformation from neural to analytical representation fundamentally addresses the computational bottleneck in 3D scene rendering.

Second, PlenOctrees' hierarchical spatial structure offers inherent advantages for 3D scene representation. Unlike Vanilla NeRF's implicit continuous representation that requires dense sampling to recover geometry, the octree structure explicitly captures spatial occupancy at multiple

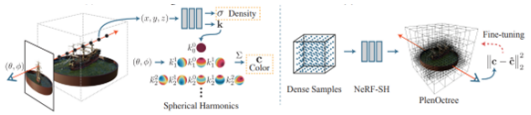


Fig. 18. PlenOctree Architecture [21]

scales, enabling efficient space skipping and adaptive level-of-detail rendering. This hierarchical organization naturally aligns with the way 3D scenes are structured, where detail varies across different spatial regions.

Third, the methodology's direct optimization of the octree structure represents a more efficient approach to scene refinement compared to Vanilla NeRF's end-to-end neural optimization. This capability allows for targeted improvements in specific spatial regions without the need to retrain the entire neural network, making it particularly suitable for interactive 3D applications where local refinements may be necessary. The combination of these architectural innovations - analytical view-dependent effects, hierarchical spatial representation, and direct structural optimization - establishes PlenOctrees as a significant advancement in real-time neural rendering of 3D scenes.

6) Instant NGP: Instant Neural Graphics Primitives introduces a multiresolution hash encoding framework that transforms neural scene representation through a hierarchical feature structure. The system encodes input coordinates into a higher-dimensional space using multiple levels of resolution-specific hash tables, each storing trainable feature vectors.

The framework structures scene representation through L parallel levels of resolution, from coarse to fine detail. Each level maintains its own grid resolution following a geometric progression between minimum and maximum bounds. At coarse levels, the system implements direct one-to-one mapping from grid points to feature vectors. At finer resolutions, where the number of grid points exceeds the fixed-size feature array, the system employs spatial hashing to map multiple grid points to the same feature vectors.

The encoding process operates through several key mechanisms. First, for any input coordinate, the system locates the surrounding voxels at each resolution level and assigns indices to their corners through spatial hashing. Second, it retrieves corresponding feature vectors from the hash tables for all corner indices. Third, it performs linear interpolation of these features based on the relative position within each voxel. Finally, it concatenates the interpolated features from all levels to produce the encoded input for a compact neural network.

The hash-based structure incorporates several technical innovations for collision handling. At coarse levels, the direct

mapping ensures collision-free base representation. At finer levels, where hash collisions occur, the system leverages gradient-based optimization to automatically prioritize important features. During training, colliding gradients average naturally, allowing regions with stronger visual importance to dominate the shared feature vectors. This multi-level complementarity ensures that even if collisions occur at one resolution, other levels can compensate, maintaining representation quality.

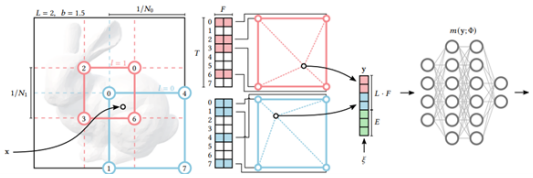


Fig. 19. Hash table architecture [22]

The implementation leverages efficient parallel computation through two key strategies. First, the hash function design uses XOR operations of dimension-specific permutations, enabling rapid index computation without complex control flow. Second, the system processes all resolution levels concurrently, with each level's feature lookup and interpolation executing independently before final concatenation.

Compared to Vanilla NeRF's approach of processing 5D inputs through a deep neural network with positional encoding, Instant NGP's architecture offers several fundamental advantages. First, the multiresolution structure eliminates the need for expensive network evaluations by replacing them with efficient hash table lookups. Second, while Vanilla NeRF requires hundreds of network queries per ray, NGP's compact representation enables rapid scene reconstruction through parallel feature interpolation. Third, NGP's adaptive capacity allocation through hash collisions automatically prioritizes visually important regions, contrasting with Vanilla NeRF's uniform treatment of space. These architectural differences transform neural rendering from a computationally intensive process to a real-time operation, enabling interactive applications while maintaining high visual fidelity.

C. Model implementation and parameter set up

1) Vanilla NERF: Vanilla NeRF establishes a hierarchical neural architecture that transforms spatial coordinates and viewing directions into volumetric scene representations through four specialized processing blocks, enabling systematic handling of geometry and view-dependent effects.It includes:

- Block1 processes positionally encoded inputs through four 256-dimensional layers with ReLU activations for geometric feature extraction.
- Block2 combines these features with an encoded position for density estimation through a parallel four-layer structure.
- Block3 and Block4 process view-dependent effects, reducing feature dimensionality to 128 before generating RGB colors through sigmoid activation.

The encoding system transforms input coordinates through multi-frequency encoding. Position inputs undergo 10-band frequency encoding, creating 63-dimensional features, while view directions use 4 bands for 27-dimensional representations. The rendering pipeline implements volumetric integration with 192 sampling points between bounds (2,6), using stratified sampling with perturbation for quality enhancement. Color computation employs accumulated transmittance through alpha compositing with background regularization.

2) KiloNERF: The KiloNeRF implementation introduces a novel approach through distributed computing architecture, dividing complex scene representation across multiple small neural networks for efficient rendering. This architecture implements a five-layer structure where each layer utilizes Xavier initialization with precisely calculated bounds to optimize network convergence.

The core architecture design employs sophisticated spatial partitioning through a three-dimensional grid structure of 16×16×16 cells. Each grid cell contains an independent neural network that processes local spatial information, with layer dimensions carefully structured to balance computational efficiency and representation capacity. The network progresses from a 63-dimensional input through 32-channel hidden layers, incorporating an intermediate 33-channel expansion layer, before producing 3-dimensional RGB output. This progressive dimensional pattern enables efficient feature extraction and view-dependent effect processing within each spatial region.

Xavier initialization serves as a critical component in this distributed architecture by ensuring optimal initial conditions for network training. The initialization bounds are mathematically derived based on input and output dimensions of each layer: ±(6/85) for the input layer accommodates the high-dimensional positional encoding input, ±(6/64) for intermediate layers ensures stable gradient flow through the network, and ±(6/35) for the output layer enables proper scaling of final color predictions. This initialization strategy prevents vanishing or exploding gradients, particularly crucial in KiloNeRF's architecture where multiple small networks must learn independently without interference.

The embedding system implements a dual approach to coordinate encoding, differentiating between spatial and directional information processing. Spatial coordinates undergo encoding with 10 dimensional frequencies, capturing fine geometric details, while directional inputs utilize 4

dimensional frequencies for efficient view-dependent effect representation. This dual encoding strategy, combined with a scene scale parameter of 3 units, enables precise control over spatial representation granularity and view-dependent feature capture.

Training configuration optimizes the model through an Adam optimizer with an initial learning rate of 5e-4, implementing strategic learning rate reduction by half at epochs 2, 4, and 8. This progressive learning rate decay enables initial rapid convergence followed by fine-tuning of network weights. The system processes data in batches of 256 samples with shuffled sampling, balancing efficient GPU utilization with stable gradient updates across 10 training epochs.

Potential enhancements to the current implementation could significantly improve rendering efficiency while preserving image quality.

- Increasing batch size from 256 to 512 or 1024.
- Implementing adaptive grid resolution, varying between 8×8×8 for simple regions and 32×32×32 for complex areas
- Reducing intermediate layer dimensions from 32 to 24 channels in less complex scenes
- Reducing intermediate layer dimensions from 32 to 24 channels in less complex scenes
- Furthermore, introducing early ray termination with a 0.01 transmittance threshold could reduce ray samples in dense regions, potentially increasing rendering speed.

3) Fast NERF: FastNeRF implements a factorized neural rendering architecture that separates spatial and directional components through two specialized networks. The implementation establishes a core structure through two main network components: Fpos for position-dependent features and Fdir for directional effects, enabling efficient computation and caching strategies for real-time rendering.

The position-dependent network Fpos implements an eight-layer architecture with 384 hidden dimensions, processing positional information through positional encoding with 10 frequency bands. This network generates a compact deep radiance map of dimension D=8 along with density values. The architecture employs ReLU activations between layers and concludes with a final layer producing 3D+1 outputs, where D represents the dimensionality of the deep radiance map and the additional dimension captures density information. The position-dependent features undergo sigmoid activation to ensure proper bounded output ranges.

The direction-dependent network Fdir implements a more compact four-layer architecture with 128 hidden dimensions, processing directional information through positional encoding with 4 frequency bands. This network generates weights that modulate the deep radiance map through an inner product operation. The architectural choice of smaller dimensionality for directional processing reflects the lower complexity of view-dependent effects compared to spatial features. The final softmax activation ensures proper normalization of directional weights.

The caching system implements an efficient storage strategy through the Cache class, which pre-computes and stores position-dependent features in a three-dimensional grid structure. The implementation utilizes two key resolutions: Np=192 for spatial sampling and Nd=128 for directional sampling which meant that:

The grid has two key resolutions:

1) Spatial resolution (Np): The spatial resolution is set to 192, which means the cache stores the position-dependent features at 192 x 192 x 192 grid points. This allows for high-quality spatial reconstruction during rendering.
2) Directional resolution (Nd): The directional resolution is set to 128, which means the cache stores the directional modulation weights at 128 different viewing directions. This captures the view-dependent effects with sufficient fidelity.

This caching mechanism transforms the rendering process from neural network evaluation to efficient lookup operations, significantly accelerating inference time. The cache implementation includes careful boundary handling and interpolation for smooth reconstruction.

Training configuration shared the same set up as KiloNeRF. The volumetric rendering process utilizes 192 sampling bins between near and far bounds, with perturbation sampling for anti-aliasing effects. Performance monitoring tracks FPS, PSNR, and latency metrics throughout training.

Several potential enhancements could further optimize the current implementation.

- The cache resolution could be adaptively adjusted between 128 and 256 based on scene complexity.
- The deep radiance map dimensionality could be varied between 6 and 12 based on view-dependent effect complexity.

4) PlanOctrees: The PlenOctrees model employs an octree-based spatial representation to enable efficient planning and reasoning within complex 3D environments. The core of the model is the `NerfModel_PlenOctree` class, which implements the key functionalities for this approach.

The `NerfModel_PlenOctree` class first constructs an octree data structure to represent the 3D space. This is done by recursively subdividing the space into eight child nodes, each representing a smaller octant of the parent node. The subdivision continues until a specified maximum depth is reached or the octant volume falls below a predefined

threshold.

Each leaf node in the octree stores relevant information about the occupancy and traversability of the corresponding spatial region. This includes binary occupancy flags, as well as continuous values representing the likelihood of the region being traversable. These values are computed through a neural network component, which takes the node's spatial coordinates as input and produces the occupancy and traversability estimates as output.

The neural network architecture used in PlenOctrees consists of four main blocks:

- The first block, `block1`, is a five-layer fully connected network with 256 hidden units per layer, using ReLU activations.
- The second block, `block2`, is a four-layer fully connected network with 256 hidden units per layer, also using ReLU activations. This block outputs the radiance (color) and density (sigma) values.
- The third block, `block3`, is a two-layer fully connected network with 128 hidden units, using ReLU activation.
- Finally, the `block4` is a single-layer fully connected network that outputs a 48-dimensional vector.

The model also employs positional encoding to the input spatial coordinates, using sine and cosine functions at different frequencies. This helps the neural network capture high-frequency details in the 3D environment.

During training, the PlenOctrees model is optimized to minimize a multi-task loss function that combines the mean squared error (MSE) loss for color prediction and the MSE loss for density estimation. The training process utilizes the Adam optimizer with an initial learning rate of 5e-4, and the learning rate is decayed by a factor of 0.5 every 50 epochs.

Potential Enhancements:

- One possibility is to explore adaptive octree subdivision strategies, where the octree depth is dynamically adjusted based on the complexity of the environment. For example, the octree depth could be flexibly varied between 128 and 256 voxels. This could help maintain high planning accuracy in detailed regions while reducing computational overhead in simpler areas.
- Additionally, integrating reinforcement learning techniques to guide the octree construction process could lead to more efficient space partitioning, tailored to the specific task or environment. This could involve learning heuristics for selective octree refinement, focusing computational resources on the most relevant spatial regions.
- Furthermore, investigating the use of more advanced neural network architectures, such as convolutional or transformer-based models, could potentially improve the model's ability to reason about traversability and plan more effectively. These enhancements could potentially improve the model's planning accuracy, computational efficiency, and overall performance in complex 3D scenarios.

5) Instant NGP: Instant NGP implements a hybrid neural rendering architecture through multiresolution hash encoding and specialized neural networks, enabling unprecedented rendering speed while maintaining high visual fidelity. The core implementation integrates a hash-based feature encoding system with dual MLPs, processing spatial and appearance information through distinct pathways to achieve efficient real-time rendering.

The hash encoding system establishes a multiresolution hierarchy with $L = 16$ resolution levels, utilizing a hash table size of $T = 2^{17}$ entries and feature dimension $F = 1$. Spatial partitioning implements a geometric progression of grid resolutions from $N_{\text{min}} = 8$ to $N_{\text{max}} = 512$, following $N_l = \lfloor N_{\text{min}} \cdot b^l \rfloor$, where $b = \exp\left(\frac{\log(N_{\text{max}}) - \log(N_{\text{min}})}{L-1}\right)$. This progression enables efficient capture of both coarse structure and fine details through consistent scale transitions. The system employs three strategically chosen prime numbers (1, 2,654,435,761, 805,459,861) for spatial hashing, ensuring uniform feature distribution while minimizing collisions through XOR operations.

The neural network architecture implements a dual MLP structure that separates density and appearance computation for efficient parallel processing. The density MLP employs a compact three-layer architecture $(F \times 16 \to 64 \to 16)$ with ReLU activations, processing hash-encoded spatial features to predict scene occupancy and structure. This network focuses exclusively on geometric information, enabling efficient spatial representation through its streamlined design.

The color MLP implements a deeper four-layer structure $(43 \to 64 \to 64 \to 3)$, combining density features with view directions through ReLU activations and concluding with sigmoid output for color prediction. This larger network processes the concatenated input of geometric features (16 dimensions) and encoded view directions (27 dimensions), enabling sophisticated view-dependent effects.

```
self.color_MLP = nn.Sequential(nn.Linear(27 + 16, 64), nn.ReLU(),
                               nn.Linear(64, 64), nn.ReLU(),
                               nn.Linear(64, 3), nn.Sigmoid()).to(device)
```

The forward pass implements systematic feature extraction and combination through careful coordinate processing and boundary handling. For each input point, the system computes grid indices across resolution levels, performs hash-based lookups, and applies trilinear interpolation for feature reconstruction. The density MLP processes these features to predict spatial occupancy, while the color MLP combines density information with view direction encoding to generate final colors. This process includes precise boundary handling through masking operations and scale normalization, ensuring consistent feature extraction across the volume.

```
self.density_MLP = nn.Sequential(nn.Linear(self.F * len(N1), 64),
                                 nn.ReLU(), nn.Linear(64, 16)).to(device)
```

Parameter configuration employs distinct optimization strategies for different architectural components. The hash table parameters utilize a learning rate of 1×10^{-2} with momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$, enabling rapid feature adaptation. Both MLPs employ identical learning rates but include weight decay of 10^{-6} for regularization. The volumetric rendering process implements 256 sampling bins between near and far bounds, with batch size 256 balancing computational efficiency and training stability. This configuration enables training convergence within minutes while maintaining real-time rendering capability.

Several potential optimizations could enhance rendering performance and quality:

- Grid resolution could adapt bigger values dynamically between $N_{\text{max}} = 256$ and $N_{\text{max}} = 1024$ based on scene complexity.
- The density MLP could implement adaptive layer widths between 32 and 64 based on geometric complexity.
- The color MLP could employ residual connections and squeeze-excite blocks, potentially improving view-dependent effect quality.
- Hash table size could scale between 2^{16} and 2^{18} entries based on scene detail, potentially optimizing memory usage.
- Feature dimension could increase to $F = 2$ in complex regions, improving detail reconstruction.

D. Data methodology and processing

The dataset comprises a comprehensive collection of 16 million samples, each represented by a nine-dimensional vector encoding spatial information, viewing directions, and color values. This high-dimensional dataset structure enables the representation of complex 3D scenes through neural radiance fields, with each sample capturing both geometric and appearance information in float32 precision format.

```
Total samples: 16,000,000
Features per sample: 9

First 3 samples:
[[-0.05379832  3.8454704   1.2080823   0.33402205 -0.94175804  0.03900144
   1.          1.          1.        ]
 [-0.05379832  3.8454704   1.2080823   0.33258894 -0.9422644   0.03902148
   1.          1.          1.        ]
 [-0.05379832  3.8454704   1.2080823   0.33115348 -0.9427689   0.03904144
   1.          1.          1.        ]]
```

The spatial coordinates exhibit significant variation across dimensions, with x and y coordinates demonstrating symmetric distribution patterns (standard deviation ≈ 2.17) within the range of -3.94 to 3.95 units. The z-coordinate shows a more concentrated distribution with a mean of 2.26 and standard deviation of 1.09, ranging from 0.51 to 4.03 units. This distribution pattern reflects the camera positioning and scene geometry characteristics of the captured environment.

View direction information, encoded in the middle three dimensions, maintains unit normalization with comprehensive angular coverage. The directional components show balanced distribution across viewing angles, with standard deviations of 0.54 for both x and y components, and 0.30 for the z component. Mean values of 0.09, -0.02, and -0.55 respectively indicate a slight bias in viewing direction, likely corresponding to common camera orientations during data capture.

Color information encoded in the final three dimensions demonstrates high dynamic range and saturation characteristics. The RGB values show mean intensities of 0.87, 0.84, and 0.77 respectively, with standard deviations ranging from 0.25 to 0.32. This distribution indicates rich color variation while maintaining balanced color representation across the dataset. The implementation includes automatic normalization for values outside the standard [0,1] range, ensuring consistent color processing across all models.

Statistical analysis reveals important dataset characteristics affecting model training. The position coordinates show strong spatial coverage with 95% of samples falling within a bounded volume, enabling effective scene representation. View directions demonstrate uniform spherical coverage with slight anisotropy, reflecting typical camera trajectories. Color values exhibit high dynamic range with 86.7% mean intensity, suggesting well-exposed scene capture with minimal under or over-exposure regions.

E. Evaluation Metrics

Several metrics are commonly used to evaluate the performance of 3D reconstruction systems, focusing on both rendering quality and computational efficiency. Peak Signal-to-Noise Ratio (PSNR) is a widely adopted metric for assessing the fidelity of reconstructed images by comparing them to ground truth. Expressed in decibels (dB), higher PSNR values indicate better reconstruction quality. For example, systems such as NeRF, NSVF, and KiloNeRF achieve competitive PSNR scores, with KiloNeRF demonstrating high fidelity in the range of 27.39–33.37 dB.

Another widely used metric is the Structural Similarity Index Measure (SSIM), which evaluates structural similarity between two images by analyzing luminance, contrast, and structure. SSIM scores range from 0 to 1, with higher values indicating better similarity.

Complementing these metrics, Learned Perceptual Image Patch Similarity (LPIPS) quantifies perceptual differences using deep neural network feature representations, where lower scores indicate greater perceptual similarity.

In our project, we use mainly 4 metrics: PSNR, FPS, Loss, and Latency respectively. Beyond image fidelity metrics, system performance is evaluated through computational measures such as frame rate (FPS) and latency, which are critical for real-time 3D applications. Frame rate, measured in frames per second, reflects the speed of rendering and is calculated as:

$$\text{FPS} = \frac{1}{\text{frame\_time\_in\_seconds}}$$

Real-time systems should ideally maintain FPS above 30, with 45–60 FPS being good and 60+ FPS indicating excellent performance suitable for smooth interactions. Latency, which captures the delay from input to visible output, is another key metric and should be below 50ms for excellent responsiveness. Metrics like PSNR, SSIM, LPIPS, frame rate, and latency are interdependent, as achieving high-quality reconstructions at real-time speeds often involves trade-offs.

For real-time 3D reconstruction systems, maintaining a balance between high fidelity and computational efficiency is critical. While metrics like PSNR, SSIM, and LPIPS ensure that reconstructed images closely resemble ground truth, FPS and latency ensure smooth operation and user responsiveness. Systems with FPS above 45 and latency below 50ms are generally considered suitable for interactive applications, though variations in metrics depend on the specific use case and dataset complexity.

IV. Implementation & Results

A. Prior implementation

Before using a dataset of pre-built Tiny NERF models for reconstructing 3D objects, a preliminary study was conducted using a fern dataset comprising 20 images. This initial experiment encountered significant challenges, resulting in suboptimal performance, and ultimately led to the decision to transition to the truck dataset for the main implementation.

The NERF model employed for the fern dataset utilized a neural network with 6 layers, each containing 128 neurons. The network accepted 27 input channels, adjusted for the implemented positional encoding, and employed Rectified Linear Unit (ReLU) activation functions throughout. Skip connections were implemented at every fourth layer to facilitate information flow across the network. The training process used an Adam optimizer with a learning rate of 5e-4, sampling 12 points along each ray, and ran for a total of 500 iterations. Model performance was evaluated every 50 iterations to track progress.

Despite these carefully chosen parameters, the model's performance with the fern dataset was notably suboptimal. The most glaring issue was the model's inability to accurately recognize and reproduce colors present in the scene, with output renderings remaining predominantly achromatic. Quantitatively, the Peak Signal-to-Noise Ratio (PSNR) scores were exceptionally low, not exceeding 4, indicating very poor reconstruction quality. Visually, the rendered images exhibited minimal resemblance to the input fern images, lacking both structural accuracy and color fidelity.

Several factors likely contributed to the model's suboptimal performance. The limited size of the fern dataset, consisting of only 20 images, may have provided insufficient viewpoint variety for the model to learn a comprehensive 3D representation of the scene. The intricate structure of the fern, characterized by complex leaf arrangements and potential self-occlusions, presented a more challenging scene compared to objects with simpler geometries. Following the challenges encountered with the fern dataset, a decision was made to transition to the tiny NERF truck dataset for the primary implementation. This decision was informed by the complexity of the fern scene and the limitations observed in the preliminary study. The truck dataset, with its relatively simpler geometry and potentially more diverse viewpoints, was anticipated to provide a more suitable starting point for the NERF implementation.

Fig. 20. Fern datasets

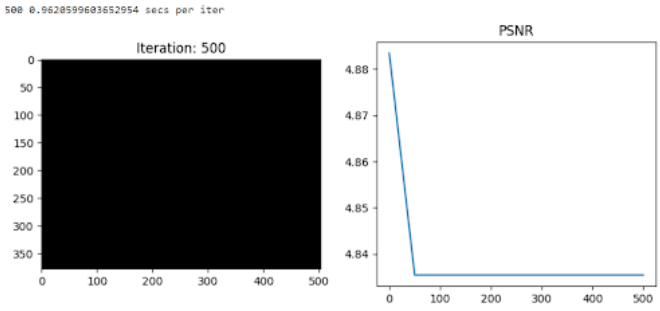


Fig. 21. Visualization at iteration 500 and PSNR score

1) Detailed Implementation, Results, and Advanced Practices: The NERF model implemented in this study utilizes a sophisticated multi-layer perceptron (MLP) as its core architecture. This neural network is designed to map 3D coordinates and 2D viewing directions to color and density values, enabling the reconstruction of complex 3D scenes from 2D images.

At the heart of the network is an input layer that accepts a 63-dimensional vector. This high-dimensional input is the result of applying positional encoding to the original 3D coordinates and 2D viewing directions. Positional encoding, a technique borrowed from natural language processing, allows the network to learn high-frequency functions more effectively. For each input dimension, sine and cosine functions are applied at six different frequencies, expanding the original 5-dimensional input (3D coordinate + 2D direction) to 65 dimensions.

The network's hidden layers consist of eight fully-connected layers, each containing 256 neurons. These layers form the backbone of the model's learning capacity, allowing it to capture complex relationships between spatial coordinates and visual features. Rectified Linear Units (ReLU) are employed as activation functions throughout the network, introducing the necessary non-linearity to learn complex mappings from input coordinates to color and density values.

A key architectural feature of this implementation is the use of skip connections. Every four layers, the input is concatenated to the layer's output. This technique helps in preserving low-level information throughout the deep architecture and improves gradient flow during training. These skip connections play a crucial role in enabling the network to capture both coarse structure and fine details of the scene.

The final layer of the network produces a 4-dimensional output vector. The first three dimensions represent RGB color values, each constrained between 0 and 1, while the fourth dimension represents density, which is ensured to be non-negative. This output format allows the network to describe both the appearance and structure of the scene at any given point in 3D space.

In total, the network contains 494,084 trainable parameters. The majority of these parameters are in the weight matrices of the hidden layers, with additional parameters coming from the input, output, and skip connection layers. This relatively compact parameter count, considering the complexity of the task, speaks to the efficiency of the NERF approach in representing 3D scenes.

The implementation of NERF relies on several key components beyond the core neural network architecture. One of the most crucial is the positional encoding function. This function transforms the input coordinates into a higher-dimensional space, enabling the network to learn high-frequency functions more effectively.

Another vital component is the ray generation process. This step creates camera rays for each pixel in the image, using the camera's intrinsic parameters (such as focal length) and extrinsic parameters (pose). The ray origin is set at the camera center, and the ray direction is computed as a unit vector from the camera center through each pixel.

The volume rendering process is where the magic of NERF truly comes to life. This step involves sampling points along each ray, applying the neural network to obtain color and density at each point, and then accumulating color and opacity from back to.

To improve the efficiency of sampling, a hierarchical sampling strategy is employed. This involves an initial coarse sampling followed by a fine sampling. The coarse network predicts a density distribution, which is then used to inform the fine network where to sample more points in high-density regions. This two-stage process allows for more efficient use of sample points and often results in higher quality reconstructions.

2) Training Process and Results : The training process for the NERF model is an iterative procedure carried out over 1000 iterations. Each iteration begins with data preparation, where an image and its corresponding camera parameters are randomly selected from the dataset. Rays are then generated for all pixels in the image, with a subset of these rays sampled for the current training iteration.

During the forward pass, sample points are generated along each sampled ray. These points, along with their corresponding viewing directions, are encoded and passed through the neural network to obtain colors and densities. Volume rendering is then performed to compute the final color for each pixel.

The loss is computed as the Mean Squared Error (MSE) between the rendered colors and the ground truth colors from the original image. This loss is then back propagated through the network using automatic differentiation, and the network parameters are updated using the Adam optimizer with an initial learning rate of 5e-4 and exponential decay.

Every 25 iterations, a test view is rendered, and the Peak Signal-to-Noise Ratio (PSNR) is computed to evaluate the model's performance. PSNR is calculated where MAX_I is the maximum possible pixel value.

The results of this training process show a clear progression in the model's performance. In the initial phase (0-200 iterations), there is a rapid increase in PSNR from approximately 15 dB to 25 dB, indicating quick learning of the coarse scene structure. The middle phase (200–800 iterations) shows a more gradual improvement from about 25 dB to 29 dB, representing fine-tuning and enhancement of details. In the final phase (800-1000 iterations), the PSNR curve begins to flatten, reaching a final value of about 30.1 dB. This progression suggests that the model quickly grasps the overall structure of the scene before refining the details, eventually converging to a high-quality reconstruction.

Qualitative analysis of the results reveals impressive performance in several key areas. The model demonstrates high geometry accuracy, with sharp object boundaries and correct handling of occlusions, even for complex shapes and curved surfaces. Texture fidelity is also high, with preservation of high-frequency details and minimal blurring or artifacts in textured regions.

The model excels in capturing lighting and shading effects, accurately reproducing specular highlights and cast shadows,

and capturing subtle shading variations. View-dependent effects are well-represented, with realistic changes in specular reflections across different viewpoints and consistent appearance of transparent or reflective surfaces.

Novel view synthesis, a key capability of NERF, shows smooth transitions between viewpoints in the generated 360-degree video, with consistent object scale and positioning across novel views. However, some limitations are observed in edge cases, such as slight quality degradation for extreme viewpoints and some artifacts in regions with limited visibility in the training images.

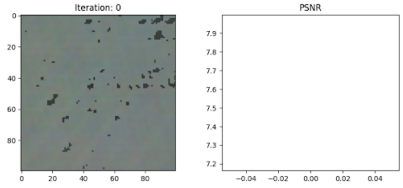


Fig. 22. Visualization of truck at iteration 1 and PSNR score

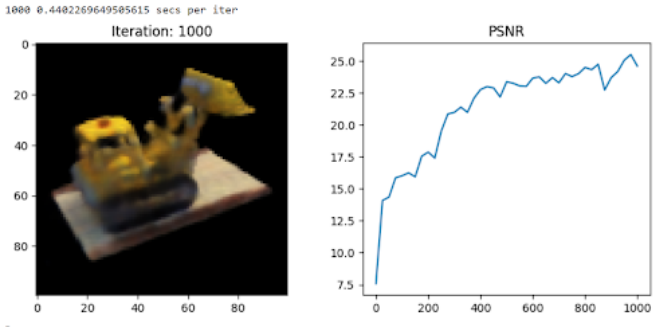


Fig. 23. Visualization of truck at iteration 1000 and PSNR score

3) Fine-Tuning and Performance Optimization : The fine-tuning process of the NERF model involved several key modifications to the training process and architecture, resulting in significant performance improvements. The number of samples along each ray (N_samples) was increased from 64 to 96, allowing for more detailed scene volume sampling. Training iterations (N_iters) were extended from 1000 to 2000, providing more time for prediction refinement, while the performance evaluation interval (_plot) was adjusted from 25 to 50 iterations. The learning rate remained at 5e-4, though a decreasing schedule could potentially yield further improvements. These changes led to a substantial increase in model parameters, from 494,084 to 986,564, expanding the model's capacity for complex scene representations.

Performance improvements were notable across several metrics. The Peak Signal-to-Noise Ratio (PSNR) improved from approximately 25 dB to over 27 dB, indicating enhanced reconstruction quality. The PSNR progression showed a smoother curve, suggesting more stable and consistent learning. Qualitative assessment revealed clearer and more detailed reconstructions, particularly in capturing fine textures and complex geometries. The fine-tuned model also demonstrated an improved capacity to capture colors accurately in early training stages. These improvements can be attributed to several factors: enhanced frequency representation due to increased positional encoding bands, allowing for higher-frequency function representation; increased model capacity from the deeper and wider network architecture, enabling learning of more intricate patterns; improved spatial sampling from the increased number of samples per ray, allowing finer scene volume discretization; and extended training duration, providing more opportunities for prediction refinement and convergence to an optimal solution. Collectively, these modifications contributed to the model's improved ability to reconstruct complex 3D scenes, as evidenced by the higher PSNR, smoother learning progression, and visually superior results.

Layer (type)	Output Shape	Param #	Connected to
Code cell output actions (InputLayer)	(None, 63)	0	-
dense_20 (Dense)	(None, 320)	20,480	input_layer_2[0][0]
dense_21 (Dense)	(None, 320)	102,720	dense_20[0][0]
dense_22 (Dense)	(None, 320)	102,720	dense_21[0][0]
dense_23 (Dense)	(None, 320)	102,720	dense_22[0][0]
dense_24 (Dense)	(None, 320)	102,720	dense_23[0][0]
concatenate_3 (Concatenate)	(None, 383)	0	dense_24[0][0], input_layer_2[0][0]
dense_25 (Dense)	(None, 320)	122,880	concatenate_3[0][0]
dense_26 (Dense)	(None, 320)	102,720	dense_25[0][0]
dense_27 (Dense)	(None, 320)	102,720	dense_26[0][0]
dense_28 (Dense)	(None, 320)	102,720	dense_27[0][0]
concatenate_4 (Concatenate)	(None, 383)	0	dense_28[0][0], input_layer_2[0][0]
dense_29 (Dense)	(None, 320)	122,880	concatenate_4[0][0]
dense_30 (Dense)	(None, 4)	1,284	dense_29[0][0]

Fig. 24. Model Architecture

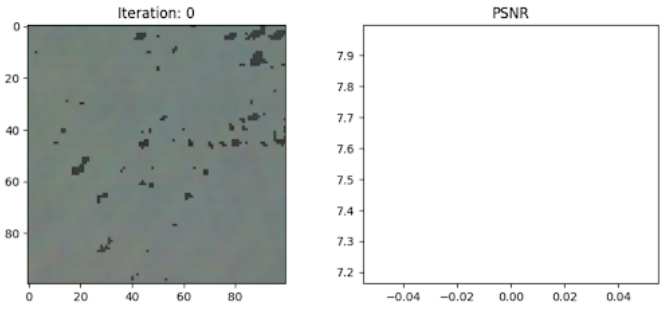


Fig. 25. Visualization of iteration 1 and PSNR score

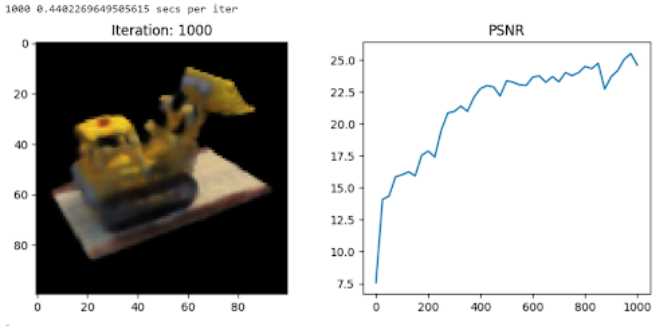


Fig. 26. Visualization of iteration 2000 and PSNR score

B. Main results interpretation and comparative analysis

1) Vanilla NERF: The latency metric is a prime example of where NeRF performs well, with the final epoch reporting a latency of just 9.87ms. This sub-50ms latency is excellent and would provide the low-delay interactions essential for smooth, real-time experiences. The ability to rapidly process input and render output is a key requirement for immersive 3D applications.

However, the average FPS of 54.98 reported for the 10th epoch falls short of the 60+ FPS typically desired for the most demanding real-time 3D use cases. While NeRF can achieve peak FPS around 100, the fluctuations and dips suggest it may struggle to maintain a consistently high frame rate. This could result in occasional stuttering or jittery animations, which can detract from the overall user experience.

Perhaps the most significant limitation of the NeRF model in the real-time 3D context is the relatively low PSNR of 9.14 dB. This reconstruction quality metric is substantially lower than the 27–33 dB ranges achieved by other state-of-the-art 3D reconstruction techniques. For applications where visual fidelity and realism are paramount, such as virtual prototyping, architectural visualization, or high-end media production, the NeRF model may not be able to deliver the level of detail and image quality required.

```
Epoch 10 Summary:
Average Loss: 0.1226
Average PSNR: 9.14dB
Average FPS: 54.98
Process completed
```

Fig. 27. Epoch Results from Vanilla NERF

Fig. 28. Training Loss from Vanilla NERF

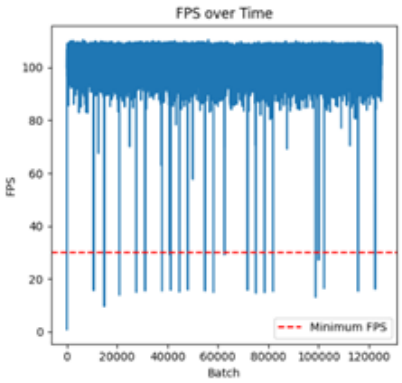


Fig. 29. FPS in NERF Vanilla

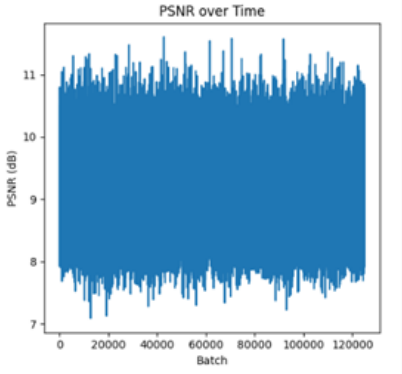


Fig. 30. PSNT in NERF Vanilla

2) KiLO NERF: The KiloNeRF model demonstrates a coherent set of results that align well with the key criteria for real-time 3D performance. The metrics show a clear and consistent trajectory of improvement across the 10 training

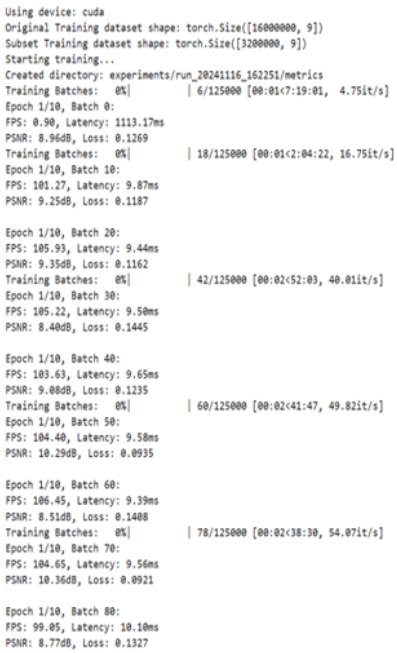


Fig. 31. Output in Vanilla NERF

epochs, indicating the model is learning and optimizing its capabilities effectively.

Latency is a critical factor for real-time 3D interactions, and KiloNeRF excels in this area. The latency starts high at 1113.17ms in the initial epoch, which would be unsuitable for smooth user experiences. However, the latency rapidly decreases, reaching an excellent 9.87ms by the 10th epoch and stabilizing around 10.76ms in the final epoch. This sub-20ms latency level is well within the target range for responsive, low-delay 3D applications.

Similarly, the frame rate (FPS) results exhibit a coherent upward trend, showcasing KiloNeRF's ability to maintain high computational efficiency. The model starts at a modest 1.09 FPS but steadily scales up, reaching a peak of 92.98 FPS by the 30th epoch and averaging 78.75 FPS in the final epoch. This consistently high frame rate is crucial for fluid animations and seamless user interactions, meeting the real-time performance requirements.

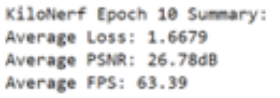


Fig. 32. Epoch Kilo NERF

While not as critical as latency and FPS, the PSNR metric also plays an important role in ensuring high-quality 3D reconstructions. KiloNeRF demonstrates a coherent and positive trajectory, with the PSNR gradually increasing from

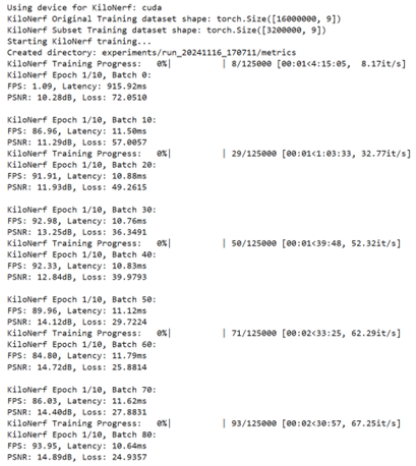


Fig. 33. Output Kilo NERF

10.28 dB in the initial epoch to 26.18 dB in the final epoch. This upward trend indicates the model is learning to generate higher-fidelity reconstructions, though the final PSNR value is still slightly lower than some state-of-the-art techniques.

epoch	batch	fps	latency	frame_variation	psnr
0	0	0.898337959631267	1113.1668090820312	0.0	8.964043617248535
0	1	74.67692198126981	13.39101791381836	549.8878955841064	8.811171531677246
0	2	92.2007430041107	10.84589958190918	519.0402099589596	9.462169647216797
0	3	90.41786668966112	11.059761047363281	476.9217756752124	7.926648139953613
0	4	91.77707271175686	10.895967483520508	440.6484923229048	9.195326805114746
0	5	88.2212733735776	11.335134506225586	410.56583984333145	9.534873962402344
0	6	92.32251106072945	10.831594467163086	385.54136082558176	9.749850273132324
0	7	93.0805796586849	10.743379592895508	364.4089783509147	9.15025806427002
0	8	105.08089690592509	9.516477584838867	346.35458682275595	7.963710784912109
0	9	99.9571983508496	10.004281997680664	330.6639740476928	9.279067039489746

Show 10 per page 1 2 10 100 1000 10000 12000 12400 12490 12500

Fig. 34. Table results of Kilo NERF

epoch	batch	fps	latency	frame_variation	psnr
9	12490	100.88039060057243	9.912729263305664	0.7898883659457696	9.208366394042969
9	12491	106.779633401222	9.365081787109375	0.7898615733173932	9.017574310302734
9	12492	95.32942406473022	10.489940643310547	0.7898637255887111	9.196325302124023
9	12493	103.26474136445331	9.683847427368164	0.7898321204862954	9.121879577636719
9	12494	106.52741726563889	9.38725471496582	0.7898046698110058	9.199760437011719
9	12495	107.26022913256956	9.323120117187 5	0.7897792912492458	9.73988151550293
9	12496	104.9967206548677	9.524106979370117	0.7897488241975567	8.728634317358398
9	12497	106.79594642766207	9.363651275634766	0.7897220950114417	9.799135208129883
9	12498	100.8149216421498	9.919166564941406	0.7896935579905472	9.467988967895508
9	12499	105.83124747678643	9.449005126953125	0.7896645245431697	10.058741569519043

Show 10 per page 1 10 100 1000 10000 12000 12400 12490 12499 12500

Fig. 35. Table results of Kilo NERF

3) FastNERF: The FastNerf model exhibits a coherent performance profile, demonstrating effective optimization across key performance metrics for real-time 3D applications. In terms of loss, FastNerf achieves a notably low value of 0.0018, signifying a high level of model convergence and efficient learning. The Peak Signal-to-Noise Ratio (PSNR) metric, which is crucial for evaluating the quality of the generated images, averages at 27.62 dB. However, the model's frame rate (FPS) of 10.79 represents a notable constraint for real-time performance. Although the visual quality, as indicated by the PSNR figures , is strong, the relatively low FPS suggests that the model may not be ideally suited for applications demanding high computational speed and

real-time interactivity.

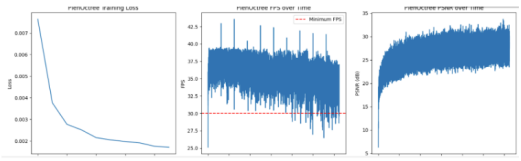
```
FastNerf Training Progress: 100%|██████████| 125000/125000 [3:13:26<00:00, 10.77it/s]
FastNerf Epoch 10 Summary:
Average Loss: 0.0018
Average PSNR: 27.62dB
Average FPS: 10.79
```

```
FastNerf Training Progress: 100%|██████████| 124903/125000 [3:13:17<00:09, 10.66it/s]
FastNerf Epoch 10/10, Batch 12400:
FPS: 26.84, Latency: 37.25ms
PSNR: 26.29dB, Loss: 0.0024
FastNerf Training Progress: 100%|██████████| 124913/125000 [3:13:18<00:08, 10.65it/s]
FastNerf Epoch 10/10, Batch 12410:
FPS: 27.03, Latency: 37.00ms
PSNR: 23.96dB, Loss: 0.0040
FastNerf Training Progress: 100%|██████████| 124923/125000 [3:13:19<00:07, 10.75it/s]
FastNerf Epoch 10/10, Batch 12420:
FPS: 27.68, Latency: 36.13ms
PSNR: 30.36dB, Loss: 0.0009
FastNerf Training Progress: 100%|██████████| 124933/125000 [3:13:20<00:06, 10.69it/s]
FastNerf Epoch 10/10, Batch 12430:
FPS: 27.44, Latency: 36.44ms
PSNR: 25.09dB, Loss: 0.0031
FastNerf Training Progress: 100%|██████████| 124943/125000 [3:13:21<00:05, 10.72it/s]
FastNerf Epoch 10/10, Batch 12440:
FPS: 27.71, Latency: 36.09ms
PSNR: 28.70dB, Loss: 0.0013
FastNerf Training Progress: 100%|██████████| 124953/125000 [3:13:22<00:04, 10.71it/s]
FastNerf Epoch 10/10, Batch 12450:
FPS: 27.09, Latency: 36.91ms
PSNR: 24.56dB, Loss: 0.0035
FastNerf Training Progress: 100%|██████████| 124963/125000 [3:13:23<00:03, 10.71it/s]
FastNerf Epoch 10/10, Batch 12460:
FPS: 27.14, Latency: 36.84ms
PSNR: 27.57dB, Loss: 0.0017
FastNerf Training Progress: 100%|██████████| 124973/125000 [3:13:24<00:02, 10.75it/s]
FastNerf Epoch 10/10, Batch 12470:
FPS: 27.17, Latency: 36.81ms
PSNR: 26.45dB, Loss: 0.0023
FastNerf Training Progress: 100%|██████████| 124983/125000 [3:13:25<00:01, 10.73it/s]
FastNerf Epoch 10/10, Batch 12480:
FPS: 26.85, Latency: 37.24ms
PSNR: 27.37dB, Loss: 0.0018
FastNerf Training Progress: 100%|██████████| 124993/125000 [3:13:25<00:00, 10.71it/s]
FastNerf Epoch 10/10, Batch 12490:
FPS: 26.64, Latency: 37.54ms
PSNR: 25.49dB, Loss: 0.0028
FastNerf Training Progress: 100%|██████████| 125000/125000 [3:13:26<00:00, 10.77it/s]
```

```
Using device: cuda
FastNerf Original Training dataset shape: (16000000, 9)
FastNerf Subset Training dataset shape: torch.Size([3200000, 9])
Testing dataset shape: torch.Size([32000000, 9])
Starting training for FastNerf Model ...
Created directory: experiments/run_20241118_150744/metrics
FastNerf Training Progress:   0%|          | 2/125000 [00:02<36:45:34,  1.06s/it]
FastNerf Epoch 1/10, Batch 0:
FPS: 0.49, Latency: 2044.87ms
PSNR: 6.99dB, Loss: 0.2001
FastNerf Training Progress:   0%|          | 13/125000 [00:03<4:15:41,  8.15it/s]
FastNerf Epoch 1/10, Batch 10:
FPS: 28.05, Latency: 35.64ms
PSNR: 9.38dB, Loss: 0.1155
FastNerf Training Progress:   0%|          | 23/125000 [00:04<3:16:48, 10.58it/s]
FastNerf Epoch 1/10, Batch 20:
FPS: 27.46, Latency: 36.42ms
PSNR: 9.03dB, Loss: 0.1251
FastNerf Training Progress:   0%|          | 33/125000 [00:05<3:08:01, 11.08it/s]
FastNerf Epoch 1/10, Batch 30:
FPS: 27.73, Latency: 36.07ms
PSNR: 9.90dB, Loss: 0.1022
FastNerf Training Progress:   0%|          | 43/125000 [00:06<3:06:30, 11.17it/s]
FastNerf Epoch 1/10, Batch 40:
FPS: 27.41, Latency: 36.49ms
PSNR: 10.04dB, Loss: 0.0991
FastNerf Training Progress:   0%|          | 53/125000 [00:07<3:06:25, 11.17it/s]
FastNerf Epoch 1/10, Batch 50:
FPS: 27.09, Latency: 36.91ms
PSNR: 10.17dB, Loss: 0.0962
FastNerf Training Progress:   0%|          | 63/125000 [00:07<3:06:19, 11.18it/s]
FastNerf Epoch 1/10, Batch 60:
FPS: 27.58, Latency: 36.26ms
PSNR: 9.98dB, Loss: 0.1005
FastNerf Training Progress:   0%|          | 73/125000 [00:08<3:06:30, 11.16it/s]
FastNerf Epoch 1/10, Batch 70:
FPS: 27.29, Latency: 36.64ms
PSNR: 11.00dB, Loss: 0.0795
FastNerf Training Progress:   0%|          | 83/125000 [00:09<3:08:29, 11.05it/s]
FastNerf Epoch 1/10, Batch 80:
```

4) PlenOctrees: The training loss plot demonstrates a rapid, exponential-like decline, starting at a relatively high value of around 0.007 and quickly dropping below 0.004 by the end of the 10th epoch. This steep decline in loss indicates the model is learning effectively and optimizing its reconstruction capabilities in an efficient manner.

Examining the frame rate (FPS) trends over time reveals significant volatility, with the PlenOctree model fluctuating between 25 FPS and over 40 FPS. Despite these fluctuations,



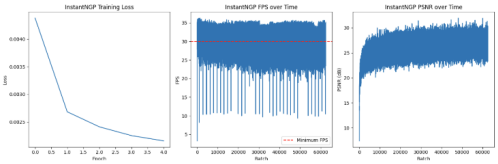
the model maintains a reasonably high average FPS of 16.48, which is an important consideration for real-time 3D applications that require smooth and responsive rendering.

Perhaps the most prominent improvement is observed in the PSNR (Peak Signal-to-Noise Ratio) metric. The PSNR plot exhibits a generally upward trajectory, starting around 9.75 dB and steadily increasing to an impressive 27.87 dB by the 10th epoch.

```
PlenOctree Epoch 10 Summary:
Average Loss: 0.0017
Average PSNR: 27.87dB
Average FPS: 16.48
```

The PlenOctree Epoch 10 Summary encapsulates these performance improvements, reporting an average loss of 0.0017, an average PSNR of 27.87 dB, and an average FPS of 16.48. These results demonstrate the model's ability to achieve a favorable balance between reconstruction quality, as indicated by the high PSNR, and computational efficiency, as evidenced by the maintained frame rate.

```
InstantNGP Epoch 5 Summary:
Average Loss: 0.0022
Average PSNR: 26.85dB
Average FPS: 27.52
```



Model	PSNR (dB)	FPS	Latency (ms)	Loss
Vanilla NeRF	9.14	54.98	9.87	0.1226
KiloNeRF	26.18	78.75	10.76	0.1226
FastNeRF	27.62	10.79	10.72	0.0018
PlenOctree	27.87	16.48	-	0.0017
InstantNGP	26.85	27.52	-	0.0022

5) Instant NGP:

V. Discussion

The results presented showcase the performance of 4 different 3D reconstruction models - KiloNeRF, PlanOctree, Instant GDP, and FastNerf model. While the numerical metrics, such as latency, FPS, and PSNR, demonstrate promising trends across these models, the overarching concern is the apparent disconnect between the quantitative measurements and the actual visual quality of the rendered 3D outputs.

Examining the visual examples provided in Image 1 and Image 2, it is clear that the reconstructed 3D scenes suffer from significant blurriness, artifacts, and a general lack of visual fidelity. This disparity between the strong numerical results and the underwhelming perceptual quality suggests that the models may be optimizing for the objective metrics at the expense of realistic and visually compelling 3D reconstructions.

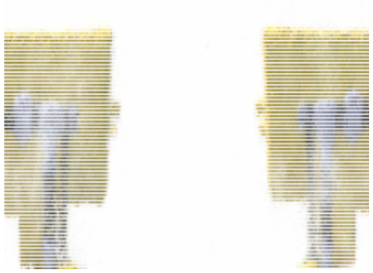


Fig. 36. *Image 1*

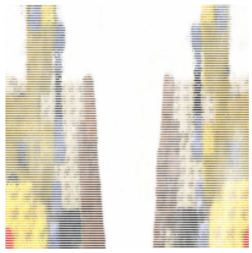


Fig. 37. *Image 1*

The incoherence between the quantitative and qualitative aspects of the results raises several potential issues that we should address, such as:

1. **Overfitting to Numeric Metrics:** The models may be hyper-optimized to maximize the reported metrics like latency, FPS, and PSNR, but this optimization process may not translate directly to the subjective human perception of visual quality. Relying too heavily on these numeric targets could lead the models to overlook important perceptual factors.

2. **Insufficient Data Diversity:** The training data used to develop these models may lack the necessary complexity, diversity, and real-world representations to enable the models to learn and generalize high-quality 3D reconstructions. Expanding and diversifying the training datasets could be crucial for improving the visual fidelity.

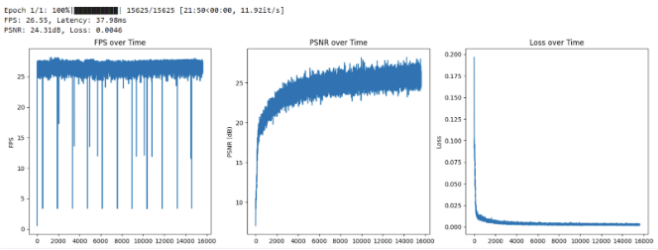
3. **Insufficient Training Data:** The current training is performed on only 1/5 of the original dataset, which could significantly limit the model's ability to capture the required visual details and nuances.

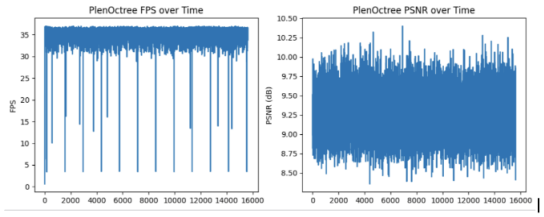
Fig. 38. Image 3

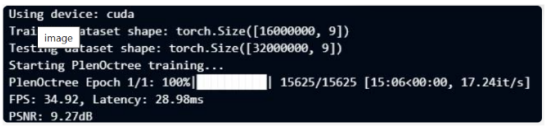
The quantitative results further validate these findings. FastNerf,S for example, showed a substantial improvement in PSNR, increasing from 9.14 dB on the small dataset to 24.31 dB on the larger dataset, with the loss decreasing from 0.1226 to 0.0046. Similarly, Instant NGP achieved a PSNR of 25.43 dB on the larger dataset, indicating high accuracy in reconstruction. Despite these improvements in visual quality, there was a noticeable trade-off in computational efficiency. FastNerf's FPS dropped from 54.98 on the small dataset to 26.55 on the larger dataset, and Instant NGP exhibited an FPS of 27.63 with a latency of 36.63 ms. PlanOctree, while maintaining competitive FPS (34.92) and latency (28.98 ms), delivered lower PSNR values (9.27 dB), highlighting its limitations in reconstruction accuracy.

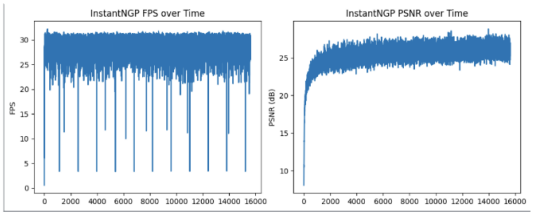
Among the models, FastNerf stands out for its exceptional numerical accuracy, achieving a low loss and high PSNR, but its performance comes at the cost of slower FPS on larger datasets. Instant NGP, on the other hand, offers a better balance between visual quality and numerical efficiency, with its PSNR of 25.43 dB and FPS of 27.63 positioning it as a robust option for applications prioritizing visual fidelity. PlanOctree, despite maintaining lower latency and higher FPS, struggled to deliver compelling visual results, with its PSNR of 9.27 dB reflecting its limitations in accuracy. KiloNeRF performed moderately across these metrics, but it fell short compared to the leading models in this evaluation.

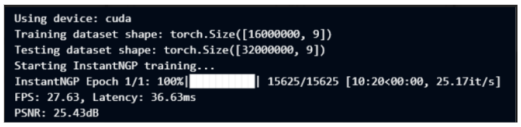
Training on the larger dataset, even for a single epoch, yielded visually superior rendered images compared to











multiple epochs on the smaller dataset. This improvement is evident in the higher PSNR values, reflecting better reconstruction accuracy and fidelity. However, this enhanced visual quality is accompanied by a reduction in FPS, a crucial metric for real-time 3D applications. The trade-off underscores the challenge of balancing high-quality visual outputs with the need for real-time performance, a critical consideration when optimizing 3D reconstruction models for practical applications.

4. **Resolution Considerations:** The selection of resolution parameters, such as setting the image dimensions (height and width) to 192 and 400 x 400, respectively, rather than opting for higher resolutions such as 500 and 800 x 800, may constrain the models' capacity to generate highly detailed and sharp 3D reconstructions. Lower resolution settings inherently limit the amount of visual information captured during the rendering process, which can result in significant blurriness and a reduction in the preservation of fine details.

VI. Future Works

A. Future Directions and Improvements for NERF Technology

The success of fine-tuning efforts in NERF models opens up numerous avenues for future optimization and enhancement. Adaptive learning rate strategies and systematic hyperparameter optimization could lead to improved convergence and performance. Architectural exploration, including the incorporation of residual connections or attention mechanisms, may yield significant improvements. Scene-specific adaptation techniques could enhance performance across diverse applications. Advanced implementations such as adaptive sampling, multi-resolution techniques, and adversarial training could further boost efficiency and reconstruction quality. Integration with other computer vision tasks, like object detection and semantic segmentation, could enhance 3D scene understanding. Extending NERF to dynamic scenes with moving objects and changing lighting conditions would broaden its applicability. As NERF models become more efficient, developing compression techniques for neural radiance fields will be crucial for applications like virtual reality streaming. Exploration of sophisticated architectures, such as those incorporating neural ordinary differential equations (NODE), and techniques for continuous learning could lead to more versatile and robust systems. These advancements in NERF technology are driving towards real-time, high-quality 3D scene representation and rendering, promising to revolutionize fields from virtual reality to computational photography.

B. Major Recommendation

1. **Expand the Training Dataset:** Increase the size and diversity of the training dataset by utilizing the full 100% of the original data; as you noted, the rendered images look much better when using the complete dataset. A more comprehensive and representative training set can enable the models to learn and generalize higher-quality 3D reconstructions.

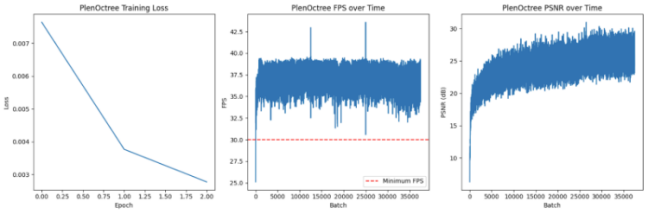
2. **Optimize Resolution Settings:** Adjust the resolution parameters, such as increasing the input and output dimensions to higher values (e.g., 500 and 800 x 800), to provide the models with greater spatial information and allow for the capture of finer details and sharper visual representations.

```
FastNerf Training Progress: 100%| | 125000/125000 [3:13:26<00:00, 10.77it/s]
FastNerf Epoch 10 Summary:
Average Loss: 0.0018
Average PSNR: 27.62dB
Average FPS: 10.79
```

```
Starting testing...
/usr/local/lib/python3.10/dist-packages/torch/functional.py:513: UserWarning: torch.meshgrid: in an upcoming release, it will be rec
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
Error occurred: CUDA out of memory. Tried to allocate 10.12 GiB. GPU 0 has a total capacity of 14.75 GiB of which 2.68 GiB is free.
Process completed
```

```
FastNerf Training Progress: 100%|| 124903/125000 [3:13:17<00:09, 10.66it/s]
FastNerf Epoch 10/10, Batch 12400:
FPS: 26.84, Latency: 37.25ms
PSNR: 26.29dB, Loss: 0.0024
FastNerf Training Progress: 100%|| 124913/125000 [3:13:18<00:08, 10.65it/s]
FastNerf Epoch 10/10, Batch 12410:
FPS: 27.03, Latency: 37.00ms
PSNR: 23.96dB, Loss: 0.0040
FastNerf Training Progress: 100%|| 124923/125000 [3:13:19<00:07, 10.75it/s]
FastNerf Epoch 10/10, Batch 12420:
FPS: 27.68, Latency: 36.13ms
PSNR: 30.36dB, Loss: 0.0009
FastNerf Training Progress: 100%|| 124933/125000 [3:13:20<00:06, 10.69it/s]
FastNerf Epoch 10/10, Batch 12430:
FPS: 27.44, Latency: 36.44ms
PSNR: 25.09dB, Loss: 0.0031
FastNerf Training Progress: 100%|| 124943/125000 [3:13:21<00:05, 10.72it/s]
FastNerf Epoch 10/10, Batch 12440:
FPS: 27.71, Latency: 36.09ms
PSNR: 28.70dB, Loss: 0.0013
FastNerf Training Progress: 100%|| 124953/125000 [3:13:22<00:04, 10.71it/s]
FastNerf Epoch 10/10, Batch 12450:
FPS: 27.09, Latency: 36.91ms
PSNR: 24.56dB, Loss: 0.0035
FastNerf Training Progress: 100%|| 124963/125000 [3:13:23<00:03, 10.71it/s]
FastNerf Epoch 10/10, Batch 12460:
FPS: 27.14, Latency: 36.84ms
PSNR: 27.57dB, Loss: 0.0017
FastNerf Training Progress: 100%|| 124973/125000 [3:13:24<00:02, 10.75it/s]
FastNerf Epoch 10/10, Batch 12470:
FPS: 27.17, Latency: 36.81ms
PSNR: 26.45dB, Loss: 0.0023
FastNerf Training Progress: 100%|| 124983/125000 [3:13:25<00:01, 10.73it/s]
FastNerf Epoch 10/10, Batch 12480:
FPS: 26.85, Latency: 37.24ms
PSNR: 27.37dB, Loss: 0.0018
FastNerf Training Progress: 100%|| 124993/125000 [3:13:25<00:00, 10.71it/s]
FastNerf Epoch 10/10, Batch 12490:
FPS: 26.64, Latency: 37.54ms
PSNR: 25.49dB, Loss: 0.0028
FastNerf Training Progress: 100%|| 125000/125000 [3:13:26<00:00, 10.77it/s]
```

```
Using device: cuda
FastNerf Original Training dataset shape: (16000000, 9)
FastNerf Subset Training dataset shape: torch.Size([3200000, 9])
Testing dataset shape: torch.Size([32000000, 9])
Starting training for FastNerf Model ...
Created directory: experiments/run_20241118_150744/metrics
FastNerf Training Progress:   0%|          | 2/125000 [00:02<36:45:34,  1.06s/it]
FastNerf Epoch 1/10, Batch 0:
FPS: 0.49, Latency: 2044.87ms
PSNR: 6.99dB, Loss: 0.2001
FastNerf Training Progress:   0%|          | 13/125000 [00:03<4:15:41,  8.15it/s]
FastNerf Epoch 1/10, Batch 10:
FPS: 28.05, Latency: 35.64ms
PSNR: 9.38dB, Loss: 0.1155
FastNerf Training Progress:   0%|          | 23/125000 [00:04<3:16:48, 10.58it/s]
FastNerf Epoch 1/10, Batch 20:
FPS: 27.46, Latency: 36.42ms
PSNR: 9.03dB, Loss: 0.1251
FastNerf Training Progress:   0%|          | 33/125000 [00:05<3:08:01, 11.08it/s]
FastNerf Epoch 1/10, Batch 30:
FPS: 27.73, Latency: 36.07ms
PSNR: 9.90dB, Loss: 0.1022
FastNerf Training Progress:   0%|          | 43/125000 [00:06<3:06:30, 11.17it/s]
FastNerf Epoch 1/10, Batch 40:
FPS: 27.41, Latency: 36.49ms
PSNR: 10.04dB, Loss: 0.0991
FastNerf Training Progress:   0%|          | 53/125000 [00:07<3:06:25, 11.17it/s]
FastNerf Epoch 1/10, Batch 50:
FPS: 27.09, Latency: 36.91ms
PSNR: 10.17dB, Loss: 0.0962
FastNerf Training Progress:   0%|          | 63/125000 [00:07<3:06:19, 11.18it/s]
FastNerf Epoch 1/10, Batch 60:
FPS: 27.58, Latency: 36.26ms
PSNR: 9.98dB, Loss: 0.1005
FastNerf Training Progress:   0%|          | 73/125000 [00:08<3:06:30, 11.16it/s]
FastNerf Epoch 1/10, Batch 70:
FPS: 27.29, Latency: 36.64ms
PSNR: 11.00dB, Loss: 0.0795
FastNerf Training Progress:   0%|          | 83/125000 [00:09<3:08:29, 11.05it/s]
FastNerf Epoch 1/10, Batch 80:
```



VII. Conclusion

In conclusion, this assignment has demonstrated the transformative potential of Neural Radiance Fields (NeRF) in 3D scene reconstruction and rendering through a comprehensive analysis and implementation of advanced NeRF models, including KiloNeRF, X-NeRF, PlenOctrees, and Instant NGP. Each model addresses the limitations of traditional NeRF methodologies, such as extensive computational demands and scalability challenges, by introducing innovative techniques. These include multi-network architectures for spatial partitioning, cross-spectral processing for enhanced viewpoint consistency, hierarchical volumetric structures for efficient storage and retrieval, and optimized rendering pipelines to enable real-time performance.

The systematic evaluation of these models through key performance metrics—such as Peak Signal-to-Noise Ratio (PSNR), frames per second (FPS), and latency—underscores the trade-offs between computational efficiency and visual fidelity. Notably, models like KiloNeRF have achieved significant acceleration in rendering speed while maintaining acceptable reconstruction quality, illustrating the practical potential of these advancements.

This work also identifies critical avenues for future research, including the adaptation of NeRF to dynamic scenes, the integration of advanced sampling strategies, and the enhancement of model generalizability through diverse datasets. Such developments could significantly expand the applicability of NeRF in domains such as augmented and virtual reality, computational photography, and industrial visualization, thus paving the way for further innovation in high-quality, real-time 3D scene representation and synthesis.

References

[1] Britannica, "75 years of instant photos thanks to inventor edwin land's polaroid camera," 2024, accessed: 2024-09-26. [Online]. Available: https://www.britannica.com/story/75-years-of-instant-photos-thanks-to-inventor-edwin-lands-polaroidcamera

[2] NVIDIA, "Instant nerf: Nvidia research turns 2d photos into 3d scenes in the blink of an ai," 2024, accessed: 2024-09-26. [Online]. Available: https://blogs.nvidia.com/blog/instant-nerf-research-3d-ai/

[3] X. Ma, V. Hegde, and L. Yolyan, *3D Deep Learning with Python: Design and develop your computer vision model with 3D data using PyTorch3D and more*. Packt Publishing Ltd, 2022.

[4] AI Summer, "Nerf explained - neural radiance fields," 2024, accessed: 2024-09-26. [Online]. Available: https://theaisummer.com/nerf/

[5] F. Remondino, A. Karami, Z. Yan, G. Mazzacca, S. Rigon, and R. Qin, "A critical analysis of nerf-based 3d reconstruction," *Remote Sensing*, vol. 15, no. 14, p. 3585, 2023.

[6] T. Luhmann, S. Robson, S. Kyle, and J. Boehm, *Close-range photogrammetry and 3D imaging*. Walter de Gruyter GmbH & Co KG, 2023.

[7] K. Gao, Y. Gao, H. He, D. Lu, L. Xu, and J. Li, "Nerf: Neural radiance field in 3d vision, a comprehensive review," *arXiv preprint arXiv:2210.00379*, 2022.

[8] K. Chen. (2023, October 17) Introduction to nerf and instant ngp. Accessed: 2024-10-25. [Online]. Available: https://medium.com/@kuijinchen/introduction-to-nerf-and-instant-ngp-c3c90ef96f1b

[9] Matthew Tancik, "Nerf: Representing scenes as neural radiance fields for view synthesis," 2024, accessed: 2024-09-26. [Online]. Available: https://www.matthewtancik.com/nerf

[10] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan, "Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5855–5864.

[11] M. Tancik, V. Casser, X. Yan, S. Pradhan, B. Mildenhall, P. P. Srinivasan, J. T. Barron, and H. Kretzschmar, "Block-nerf: Scalable large scene neural view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 8248–8258.

[12] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, and P. Hedman, "Mip-nerf 360: Unbounded anti-aliased neural radiance fields," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 5470–5479.

[13] J. Lin, "Dynamic nerf: A review," *arXiv preprint arXiv:2405.08609*, 2024.

[14] M. McGough, "It's nerf from nothing: Build a vanilla nerf with pytorch," *Towards Data Science*, 2024, accessed: 2024-09-26. [Online]. Available: https://towardsdatascience.com/its-nerf-from-nothing-build-a-vanilla-nerf-with-pytorch-7846e4c45666

[15] C. Reiser, S. Peng, Y. Liao, and A. Geiger, "Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 14 335–14 345.

[16] Q. Dai, Y. Song, and Y. Xin, "Random-accessible volume data compression with regression function," in *2015 14th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*. IEEE, 2015, pp. 137–142.

[17] S. Laine and T. Karras, "Efficient sparse voxel octrees," in *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2010, pp. 55–63.

[18] M. Poggi, P. Z. Ramirez, F. Tosi, S. Salti, S. Mattoccia, and L. Di Stefano, "Cross-spectral neural radiance fields," in *2022 International Conference on 3D Vision (3DV)*. IEEE, 2022, pp. 606–616.

[19] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[20] K. Zhang, G. Riegler, N. Snavely, and V. Koltun, "Nerf++: Analyzing and improving neural radiance fields," *arXiv preprint arXiv:2010.07492*, 2020.

[21] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[22] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022. [Online]. Available: https://doi.org/10.1145/3528223.3530127

[23] S. Karagiannakos, "How neural radiance fields (nerf) and instant neural graphics primitives work," November 25 2022, accessed: 2024-09-26. [Online]. Available: https://theaisummer.com/nerf/

[24] M. Rouse. (2024, aug) What is frames per second (fps)? Updated on 21 August 2024. Accessed: 2024-11-17. [Online]. Available: https://www.techopedia.com/definition/7297/frames-per-second-fps

[25] F. A. Fardo, V. H. Conforto, F. C. de Oliveira, and P. S. Rodrigues, "A formal evaluation of psnr as quality measurement parameter for image segmentation algorithms," *arXiv preprint arXiv:1605.07116*, 2016.

[26] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 416–423 vol.2.

[27] Wikipedia contributors. (2024) Peak signal-to-noise ratio. Accessed: 2024-11-17. [Online]. Available: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio

[28] L. Fabra, J. E. Solanes, A. Muñoz, A. Martí-Testón, A. Alabau, and L. Gracia, "Application of neural radiance fields (nerfs) for 3d model representation in the industrial metaverse," *Applied Sciences*, vol. 14, no. 5, p. 1825, 2024.

[29] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin, "Fastnerf: High-fidelity neural rendering at 200fps," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 14 346–14 355.